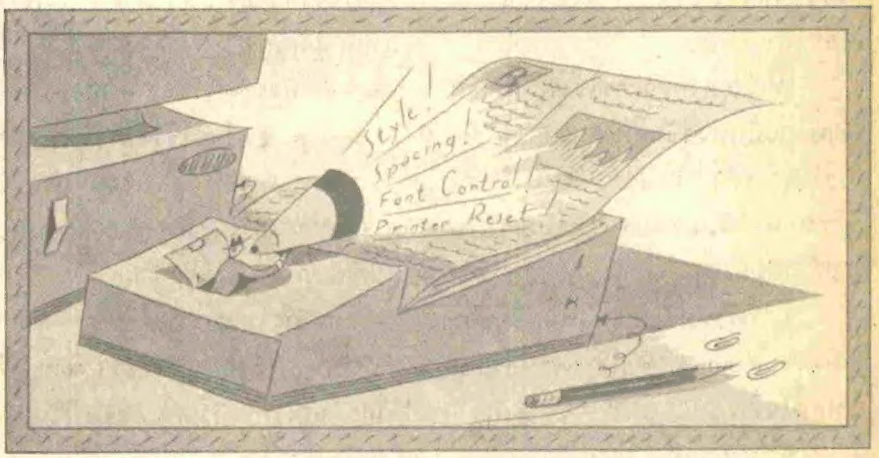
ซอฟต์แวร์
อาจหาญ สัตยารักษ์

SOFT MAIL

พิมพ์จ่าหน้าพัสดุ ด้วยภาษาไทยกราฟิก รหัส สมอ.

เรียนรู้การป้อนข้อความในโหมดกราฟิก และการแสดงข้อความกราฟิกขนาดใหญ่ ด้วยรหัส สมอ.



เพื่อให้การทำภาษาไทยรหัส สมอ. มีความสมบูรณ์ที่สุด โดยผู้อ่านจะนำไปใช้ประโยชน์ได้สูงสุด ผู้เขียนจึงเขียนบทความในเรื่องภาษาไทยกราฟิกรหัส สมอ.อีกครั้ง โดยครั้งนี้จะกล่าวถึงเรื่องสำคัญ 2 เรื่องคือ

1. การป้อนข้อความในโหมดกราฟิก ซึ่งได้เสนอเป็นยูนิตไฟล์ในชื่อ READST.TPU

2. การแสดงข้อความกราฟิกขนาดใหญ่โดยอาศัยฟอนต์ตัวหนังสือจากไฟล์ที่ใช้ในการพิมพ์ข้อความลงในกระดาษ โดยสร้างเป็นยูนิตชื่อ TISBIG.TPU

ตัวอย่างการใช้งานของยูนิตในข้อ 1 และ 2 นั้น เป็นการป้อนชื่อที่อยู่บนจอภาพ และนำข้อมูลที่ป้อนนั้นมาแสดงบนจอภาพใหม่ด้วยภาษาไทยกราฟิกขนาดใหญ่ เมื่อใช้โปรแกรมที่ลอกภาพหน้าจอให้พิมพ์ลงกระดาษก็นำไปเป็นจ่าหน้าห่อพัสดุส่งทางไปรษณีย์ได้อย่างสวยงาม ซึ่งส่วนนี้คือไฟล์ชื่อ POSTPNT.PAS

ผู้เขียนได้จัดทำเป็นซอฟต์เมล์เช่นเคย โดยมีไฟล์อื่น ๆ ที่เกี่ยวข้องมาให้อย่างครบถ้วน

ผู้อ่านที่เพิ่งมาอ่านบทความฉบับนี้ โปรดทราบว่า บทความชุดนี้ได้เขียนไปแล้ว 2 ตอน คือ ส่วนการทำภาษาไทยกราฟิก (ขนาดปกติ) บนจอภาพ และการพิมพ์ข้อความลงกระดาษ ดังนั้น กรุณาอ่านรายละเอียดต่าง ๆ ได้จากบทความในฉบับที่ผ่าน ๆ มา

ป้อนข้อความกราฟิก

เมื่อเราได้ภาษาไทยกราฟิกรหัส สมอ. (วารสารไมโครฯฉบับที่ 88) มาใช้งานแล้ว ส่วนสำคัญอีกอย่างหนึ่งที่ผู้อ่านต้องการให้ผู้เขียนเขียนถึงนอกเหนือจากการพิมพ์ลงกระดาษก็คือ การป้อนข้อความในโหมดกราฟิก ดังนั้น จึงขอกล่าวถึงการป้อนข้อความก่อนเป็นอันดับแรก

หลักการในการป้อนข้อความก็จะมีแนวคิดแบบเดียวกับการป้อนข้อความในภาษาไทยรหัสเกษตรที่เคยได้เขียนไว้ในฉบับก่อน ๆ ดังนี้

1. กำหนดตำแหน่งเริ่มต้นการป้อน โดยตำแหน่งนี้จะเป็นตำแหน่งเดียวกันกับการแสดงข้อความ จำนวนไบต์สูงสุดที่ป้อนได้ และโหมดการเริ่มต้นป้อนว่าต้องการให้เป็นภาษาไทย หรือภาษาอังกฤษ

2. แสดงเคอร์เซอร์แบบกราฟิกให้ปรากฏบนจอภาพ เพื่อให้ผู้ใช้รู้ว่ากำลังป้อนข้อความ โดยให้ขนาดเคอร์เซอร์เป็นตัวกำหนดให้ทราบว่าเป็นการป้อนภาษาไทย (สี่เหลี่ยมขนาดโต) หรือเป็นการป้อนภาษาอังกฤษ (สี่เหลี่ยมขนาดเล็ก)

3. เมื่อผู้ใช้กดแป้นเพื่อป้อนข้อความ โปรแกรมจะตรวจว่ากำลังอยู่ใน

การป้อนภาษาใด ถ้าป้อนภาษาอังกฤษก็ส่งอักขระที่ป้อนนั้นไปแสดงบนจอภาพทันที หากเป็นการป้อนภาษาไทยจะต้องนำอักขระที่ป้อนนั้นมาเทียบอักขระภาษาไทยตามตำแหน่งการป้อน เพื่อเปลี่ยนเป็นอักขระภาษาไทยเสียก่อน แล้วจึงส่งอักขระใหม่นี้ไปแสดงบนจอภาพ

เพื่อสะดวกในการใช้งาน ผู้เขียนจึงเขียนเป็นฟังก์ชันชื่อ ReadSTR ซึ่งอยู่ในยูนิตชื่อ ReadST.TPU ดังมีรายละเอียดเบื้องต้นของโปรแกรมในรูปที่ 1 มีดังนี้

1. บรรทัดที่ 2 กำหนดชื่อของยูนิตว่า ReadST ซึ่งผู้อ่านจะต้องเก็บไฟล์นี้ในชื่อ READST.PAS ด้วย

2. บรรทัดที่ 4 กำหนดยูนิตอื่น ๆ ที่ใช้งานร่วมด้วย

3. บรรทัดที่ 5 กำหนดตัวแปร Sec100 ซึ่งจะเป็นตัวแปรที่เก็บเวลาของ 1/100 วินาที เพื่อที่ผู้อ่านจะนำไปใช้งานต่อไปได้

4. บรรทัดที่ 6 ชุดคำสั่งสำหรับหาค่าของตัวแปร Sec100 ซึ่งเรียกใช้ได้ภายนอกยูนิต

5. บรรทัดที่ 7 ชุดคำสั่งสำหรับลบจอภาพ ซึ่งเรียกใช้ได้ภายนอกยูนิต

6. บรรทัดที่ 8 ฟังก์ชันสำหรับการป้อนข้อความ ซึ่งเรียกใช้ได้ภายนอกยูนิตเช่นกัน

7. บรรทัดที่ 12-16 ชุดคำสั่งสำหรับหาค่าตัวแปร Sec100 ซึ่งหาได้จากระบบเวลาของเครื่องคอมพิวเตอร์ผ่านอินเตอร์รัพต์หมายเลข 21 (ฐานสิบหก)

8. บรรทัดที่ 17-22 ส่วนของการลบจอภาพในขอบเขตที่กำหนด โดยการสั่งขนาดของกรอบภาพให้ได้ขนาดตามที่กำหนดก่อน เมื่อใช้คำสั่งลบจอภาพจึงทำให้ส่วนของจอภาพที่กำหนดนั้นถูกลบไป หลังจากนั้นก็กำหนดขนาดของจอภาพให้โตเต็มหน้าจอภาพใหม่อีกครั้ง

9. บรรทัดที่ 23-27 ส่วนการเปล่งเสียง และลบล้างค่าอักขระในตัวแปร ch เพื่อนำไปใช้ในการป้อนข้อความ คือเมื่อมีการป้อนข้อความภาษาไทยผิดอักขระวิธี เช่นป้อนสระ ุ และ ู ตามหลังวรรณยุกต์ เป็นต้น ซึ่งโปรแกรมจะเรียกใช้ชุดคำสั่งนี้เพื่อกำหนดให้การป้อนอักขระนั้นล้มเหลวไป จึงไม่มีการนำอักขระนั้นไปแสดงบนจอภาพ

ส่วนการป้อนข้อความได้เขียนเป็นฟังก์ชันด้วยเหตุผลสำคัญคือ

การเขียนเป็นฟังก์ชันจะมีรูปแบบการเรียกใช้ดูเข้าใจง่ายกว่า เช่น

```
Name: = ReadSTR(2,3,3,'T');
```

ในขณะที่ถ้าเขียนเป็นชุดคำสั่งอาจต้องเขียนดังนี้

```
ReadSTR(2,3,3,'T');
Name: = InputST;
```

ในกรณีนี้จะต้องกำหนด InputST ให้เป็นตัวแปรโกลบอลหรือเรียกใช้ว่า

```
ReadSTR(2,3,3,'T',Name);
```

โดยกรณีนี้ต้องกำหนดที่ชุดคำสั่งในลักษณะดังนี้

```
ReadSTR(x,y:word; Count: byte; TMaode:char; Var InputST: string);
```

```
1: {============ READST.PAS ============}
2: UNIT ReadST;
3: interface
4: uses crt,dos,graph,Thaitis;
5: var Sec100:word;
6: PROCEDURE FindSec100;
7: PROCEDURE ClsGR(x1,y1,x2,y2:integer);
8: FUNCTION ReadSTR(Column,Row,Count:byte; TTmode:char):String;
9: implementation

10: var InputSt:string;
11:     Regs:registers;

12: PROCEDURE FindSec100;
13: begin
14:    regs.AX:=$2C00; Intr($21,regs);
15:    Sec100:=Regs.DL;
16: end;

17: PROCEDURE ClsGR(x1,y1,x2,y2:integer);
18: begin
19:     setviewport(x1,y1,x2,y2,clipon);
20:     clearviewport;
21:     setviewport(0,0,GetMaxx,Getmaxy,clipon);
22: end;

23: PROCEDURE ClearCh;
24: begin
25:       sound(500); delay(150); nosound;
26:       ch:=#0;
27: end;
```

รูปที่ 1 ส่วนหัวของยูนิต กำหนดชุดคำสั่งที่ใช้งาน

ซึ่งถ้าพิจารณาการใช้คำสั่ง 2 แบบ คือ

```
Name:=ReadSTR(2,3,3'T');
ReadSTR(2,3,3'T'),Name);
```

จะเห็นว่าคำสั่งแบบแรกสื่อความหมายได้ดีกว่าคำสั่งแบบที่สอง ซึ่งโปรแกรมที่เขียนเป็นดังรูปที่ 2 มีรายละเอียดของฟังก์ชัน ReadSTR ดังนี้

1. บรรทัดที่ 30-40 ส่วนของตารางเทียบภาษาอังกฤษให้เป็นภาษาไทยรหัส สมอ. ซึ่งผู้เขียนได้แบ่งออกให้เห็นชัดเจนตามรหัสแอสกีภาษาอังกฤษ โดยยึดตำแหน่งแป้นพิมพ์ที่นิยมใช้ในปัจจุบันนี้เป็นเกณฑ์ โดยในส่วนของตัวเลขได้กำหนดให้แสดงเลขไทยในกรณีที่อยู่ในการป้อนข้อความภาษาไทยด้วย

2. บรรทัดที่ 41-43 ส่วนการกำหนดตัวแปรที่เกี่ยวข้อง เช่น Tmode ตรวจว่าอยู่ในการป้อนภาษาใด (Tmode เป็นจริงเมื่อป้อนภาษาไทย และเป็นเท็จเมื่อป้อนภาษาอังกฤษ)

3. บรรทัดที่ 45 กำหนดให้ตัวแปร Tmode เป็นเท็จ

4. บรรทัดที่ 46 ถ้าค่า TTmode ='T' หรือ 't' จะกำหนดให้ Tmode เป็นจริงซึ่งแสดงว่าจะเริ่มที่การป้อนข้อความภาษาไทย

5. บรรทัดที่ 48 กำหนดตัวแปร Ypix ซึ่งใช้ในการแสดงเคอร์เซอร์ให้แตกต่างกันในแต่ละภาษา

6. บรรทัดที่ 49-50 กำหนดให้แสดงเคอร์เซอร์เป็นสีหมายเลข 3 โดยใช้การวาดรูปสี่เหลี่ยมให้เห็นเป็นเคอร์เซอร์ตามขนาดที่คำนวณได้

7. บรรทัดที่ 53-58 เป็นการวนรอบแสดงเคอร์เซอร์ โดยแสดงแบบกะพริบด้วยการวาดรูปสี่เหลี่ยมสลับการลบจอภาพในขอบเขตเดียวกัน ช่วงของการกะพริบนั้นตรวจจากตัวแปร Sec100 ที่หาได้จากชุดคำสั่ง FindSec 100 จึงทำให้การกะพริบจะคงที่ ไม่ว่าจะใช้กับเครื่องที่มีความเร็วมากหรือน้อย

8. บรรทัดที่ 62-65 เป็นการเปลี่ยนโหมดการป้อนสลับไปมาระหว่างภาษาไทย และภาษาอังกฤษ โดยการใช้แป้น Esc (รหัสแอสกีเท่ากับ 27) หากผู้อ่าน

รูปที่ 2 ฟังก์ชันสำหรับป้อนข้อความภาษาไทยกราฟิกรหัส สมอ.

```
28: FUNCTION ReadSTR(Column,Row,Count:byte; TTmode:char):String;
29:  CONST
30: ThaiChr:array[33..125] of integer=
31:                (046,046,242,243,244,000,167,    {33-39}
32:    246,247,245,249,193,162,227,189,168,197,     {40-49}
33:    047,045,192,182,216,214,164,181,171,199,     {50-59}
34:    178,170,204,063,241,196,037,169,175,174,     {60-69}
35:    226,172,223,179,235,201,200,063,236,207,     {70-79}
36:    173,240,177,166,184,234,206,034,041,235,     {80-89}
37:    040,186,092,197,217,248,096,191,212,225,     {90-99}
38:    161,211,180,224,233,195,232,210,202,183,     {100-109}
39:    215,185,194,230,190,203,208,213,205,228,     {110-119}
40:    187,209,188,176,124,044);                    {120-125}
41: var xxpix,k,x:word;
42:     Ypix:byte; stck:string[4];
43:     Tmode:boolean;

44: begin
45:    Tmode:=False;
46:    if Upcase(TTmode)='T' then Tmode:=True;
47:    k:=Column; InputSt:=''; x:=0;
48:    ypix:=7; if pred(Tmode) then Ypix:=15;
49:    set_color(3); setfillstyle(1,GetColor);
50:    bar((k-1)*8+1,(Row-1)*16+Ypix,k*8,Row*16);
51:  repeat  ch:=#0;
52:   repeat
53:     repeat
54:       FindSec100;
55:       if Sec100 mod 2 = 1 then
56:            ClsGr((k-1)*8+1,(Row-1)*16+ypix,k*8,Row*16)
57:        Else bar((k-1)*8+1,(Row-1)*16+Ypix,k*8,Row*16);
58:     until keypressed;
59:       readchar;
60: until ch in [#8,#13,#27,#32..#255];
61:    ClsGr((k-1)*8+1,(Row-1)*16+ypix,k*8,Row*16);
62: if ch=#27 then
63:     begin if Tmode then Tmode:=false Else Tmode:=true;
64:          ypix:=7; if pred(Tmode) then Ypix:=15;
65:     end;
66: if tmode and (ch in [#33..#125]) then  ch:=chr(ThaiChr[ord(ch)]);
67: if (inputst[length(inputst)] in [#212..#215,#209]) and
68:             (ch in [#216,#217,#212..#215,#209]) then  ClearCh;
69: if (inputst[length(inputst)] in [#212..#215,#231..#236])
70:              and (ch in [#216,#217]) then  ClearCh;
71: if (inputst[length(inputst)] in [#216,#217])
72:         and (ch in [#216,#217,#209,#212..#215,#236]) then  ClearCh;
73: if (inputst[length(inputst)] in [#32..#127,#207,#208,
74:                #210,#224..#231,#240..#255])
```

```
75:          and (ch in [#216,#217,#209,#212..#219,#231..#237]) then  ClearCh;
76: if (ch in [#32..#255]) and (x<Count) then
77:   begin
78:       stck:='   ';
79:       Inc(x); stck[1]:=ch; xxpix:=0;
80:   if ch in [#232..#236] then
81:      begin
82:        if inputst[length(inputst)] in [#209,#212..#215] then
83:          begin StCk:=copy(inputst,length(inputst)-1,2)+ch;
84:                xxpix:=1;
85:          end;
86:        if inputst[length(inputst)]=#211 then
87:          begin
88:            stck:=copy(inputst,length(inputst)-1,1)+ch+#211;
89:            xxpix:=2;
90:          end;
91:      end;
92:       InputSt:=InputSt+ch;  OutThaiXY(k-xxpix,Row,Stck);
93:       if Ch in [#32..#208,#210,#211,#224..#230,#240..#249] then Inc(k);
94:       set_color(3); setfillstyle(1,getcolor);
95:       if count=1 then ch:=#13;
96:   end;
97:   if (ch=#8) and (x>0) then
98:    begin
99:     if inputst[length(inputst)] in [#32..#208,#210,#211,#224..#230,
100:                    #240..#249] then dec(k);
101:     if inputst[length(inputst)] in [#216,#217] then
102:                     ClsGR((k-2)*8,Row*16,(k-1)*8,Row*16+7);
103:     if inputst[length(inputst)] in [#209,#212..#215,#231] then
104:                   ClsGR((k-2)*8,(Row-1)*16,(k-1)*8,(Row-1)*16+6);
105:     if inputst[length(inputst)] in [#232..#236] then
106:       begin
107:         if inputst[length(inputst)-1] in [#209,#212..#215,#231] then
108:                      ClsGR((k-2)*8,(Row-1)*16-7,(k-1)*8,(Row-1)*16+2);
109:         if inputst[length(inputst)-1] in [#32..#208,#210,#211,#216,#217,
110:               #224..#230,#240..#249] then
111:                    ClsGR((k-2)*8,(Row-1)*16,(k-1)*8,(Row-1)*16+2);
112:       end;
113:        InputSt:=Copy(InputSt,1,Length(InputSt)-1);
114:        ClsGR((k-1)*8,(Row-1)*16,k*8,Row*16+12);
115:        Dec(x);
116:    end;
117: until ch=#13;
118:   ReadSTR:=InputSt;
119: end;
120: end.
```

รูปที่ 2 (ต่อ)

ฉนัดการใช้แป้นอื่นก็ให้กำหนดใหม่ หรืออาจกำหนดค่านี้ให้เป็นตัวแปรโกลบอลที่เปลี่ยนค่าได้จากภายนอกยูนิตก็ได้ เช่น ให้เป็น ThaiKey:= #27; ซึ่งก็จะเปลี่ยนค่าได้ตามต้องการนอกยูนิตได้

9. บรรทัดที่ 66-75 เป็นการตรวจการป้อนภาษาไทยให้ถูกอักขระวิธี

10. บรรทัดที่ 76-96 เป็นการนำอักขระที่ป้อนไปแสดงบนจอภาพ โดยตรวจในกรณีการป้อนวรรณยุกต์ตามคำสั่งบรรทัดที่ 80-90 เป็นกรณีพิเศษด้วย การเลื่อนคอลัมน์ให้ไปข้างหน้าจะกำหนดเฉพาะตัวอักขระระดับกลางเท่านั้น

11. บรรทัดที่ 97-116 ส่วนตรวจการลบข้อความด้วยแป้นลูกศรสำหรับลบข้อความ (รหัสแอสกี 8) ซึ่งจะตรวจสอบที่ส่วนท้ายของตัวแปร InputST ว่าเป็นอักขระระดับใด เพื่อที่จะได้ลบอักขระนั้นได้ถูกต้อง รวมทั้งตรวจการเลื่อนเคอร์เซอร์ถอยหลังเมื่อพบว่าอักขระท้ายสุดนั้นเป็นอักขระระดับกลาง

12. บรรทัดที่ 118 กำหนดให้ฟังก์ชัน ReadSTR มีค่าเท่ากับตัวแปร InputST

เนื่องจากภาษาปาสคาลสามารถเปลี่ยนข้อความให้เป็นตัวเลขได้อยู่แล้ว

ดังนั้น การเขียนคำสั่งเสริมหลังการเรียกใช้ฟังก์ชันนี้จะทำให้ได้ข้อมูลตัวเลข ทั้งนี้ต้องกำหนดตัวแปรรองรับให้เหมาะสมด้วย เช่น

กำหนดตัวแปรเป็น

```
var Sala y:real; ck:integer;
```

ใช้คำสั่งในการทำงาน

```
Val(ReadSTR(2,5,12,'E'),Salary,
Ck);
```

สำหรับตัวอย่างการใช้งานนั้น ผู้เขียนจะกล่าวถึงในตอนที่สามของบทความ เพราะจะขอเขียนถึงการสร้างภาษาไทยกราฟิกขนาดใหญ่เสียก่อน

สร้างภาษาไทยกราฟิกขนาดใหญ่

หลังจากที่ได้เสนอบทความเรื่องการพิมพ์อักขระแบบกราฟิกลงเครื่องพิมพ์ไปแล้ว ผู้เขียนพิจารณาเห็นว่าข้อมูลของอักขระในไฟล์ normal.p24 และ italic.p24 นั้น สามารถนำมาเขียนอักขระกราฟิกขนาดใหญ่บนจอภาพได้ ทั้งนี้เพราะว่าข้อมูลในการพิมพ์นั้นใช้ควบคุมการพิมพ์ของ 24 หัวเข็ม จึงสามารถนำมาลงจุดได้ 24 จุดในแนวดิ่งได้ รายละเอียดของข้อมูลหนึ่งอักขระมีดังนี้

1. ข้อมูลที่อยู่แถวบนคือ ข้อมูลลำดับที่หารด้วย 3 เศษ 1 คือข้อมูลลำดับที่ 1, 4, 7 52

2. ข้อมูลที่อยู่แถวกลางคือ ข้อมูลลำดับที่หารด้วย 3 เศษ 2 คือข้อมูลลำดับที่ 2, 5, 8 53

3. ข้อมูลที่อยู่แถวล่างคือ ข้อมูลลำดับที่หารด้วย 3 ลงตัว คือ ข้อมูลลำดับที่ 3, 6, 9 54

นั่นคือข้อมูลหนึ่งชุดมี 18 แนวตั้ง แต่ละแนวตั้งมีข้อมูล 3 ไบต์ (18×3= 54 ไบต์)

ถ้านำส่วนของข้อมูลเหล่านี้มาลงจุดในตำแหน่งที่ถูกต้อง ก็จะแสดงอักขระนั้นบนจอภาพในขนาดขอบเขต 16×24 ได้ โดยเราอาจจะกำหนดให้แสดงในลักษณะกำหนดตำแหน่งแน่นอนหรือให้ลงในตำแหน่งใด ๆ ตามความเป็นจริงก็ได้

เพื่อให้สามารถนำไปใช้งานกับภาษาไทยขนาดปกติได้ จึงอาศัยการเข้ากราฟิกจากยูนิต thaitis ที่ได้เขียนไปแล้วมาช่วยงาน โดยใช้ชื่อไฟล์ว่า TISBIG.PAS ซึ่งจะได้ยูนิตไฟล์ชื่อ TISBIG.TPU

โดยโปรแกรมในรูปที่ 3 จะเป็นส่วนแรกของยูนิตที่กล่าวถึงนี้ และรายละเอียดของโปรแกรมในรูปที่ 3 มีดังนี้

1. บรรทัดที่ 4-5 กำหนดยูนิตใช้งานและชุดคำสั่งที่เรียกใช้จากภายนอก

```
 1: {================ TISBIG.PAS ====================}
 2: UNIT TISBIG;
 3: interface
 4: uses graph,ThaiTIS;
 5: PROCEDURE OutBig(column,row:integer; content:string);
 6: implementation
 7: Type Font=array[1..54] of byte;
 8: var  fontP:array[1..2,0..223] of Font;
 9:      NoFont:byte;

10: PROCEDURE ReadP24;
11: var f:file; k:word;
12: begin
13:   assign(f,'Normal.p24');
14:   reset(f,1);
15:   blockread(f,fontP[1],12096);
16:   close (f);
17:   assign(f,'Italic.p24');
18:   reset(f,1);
19:   blockread(f,fontP[2],12096);
20:   close (f);
21:   NoFont:=1;
22: end;

23: PROCEDURE ShowCharBig(x,y:integer; ASCII:byte);
24: const BitVal:array[0..7] of byte=
25:           (128,64,32,16,8,4,2,1);
26:       YY:array[0..2] of byte=(16,0,8);
27: var k,B:word;
28: begin
29:  for k:=1 to 54 do
30:   begin
31:    if fontP[NoFont][ASCII-32][k] <> 0  then
32:     begin
33:       for B:=0 to 7 do
34:         if fontP[NoFont][ASCII-32][k] and BitVal[B]=BitVal[B] then
35:                              Putpixel(X,Y+B+YY[k mod 3],Getcolor);
36:     end;
37:     if k mod 3=0 then Inc(x);
38:   end;
39: end;
```

รูปที่ 3 ชุดคำสั่งอ่านข้อมูลจากไฟล์ และการแสดงอักขระหนึ่งตัวลงบนจอภาพ

ยูนิตที่เขียน

2. บรรทัดที่ 7-9 กำหนดชนิดของตัวแปรเพื่อรองรับข้อมูลของไฟล์ normal.p24, italic.p24 ซึ่งมีขนาด 54 ไบต์ต่อหนึ่งตัวอักษร และแต่ละไฟล์จะมีข้อมูลอยู่ 224 ชุด (ข้อมูลลำดับที่ 0-223) โดยขอให้ดูรายละเอียดของลำดับที่ของแต่ละอักขระในเรื่อง การพิมพ์อักขระกราฟิกรหัส สมอ. ที่ผู้เขียนได้เขียนในฉบับที่ผ่านมา ตัวแปร NoFont คือชุดของฟอนต์ข้อมูล nofont = 1 จะแสดงอักขระจากไฟล์ normal.p24 และมีค่า 2 จะใช้ฟอนต์จากไฟล์ italic.p24

3. บรรทัดที่ 10-22 เป็นการอ่านข้อมูลจากไฟล์ normal.p24 มาบรรจุในตัวแปร FontP[1] และจากไฟล์ italic.p24 มาบรรจุในตัวแปร FontP[2] พร้อมกับกำหนดให้เรียกใช้ FontP[1] เป็นการเริ่มต้น ค่าของตัวแปรนี้เปลี่ยนแปลงได้โดยการส่งรหัสควบคุมให้กับข้อความนั้น ๆ

4. บรรทัดที่ 23-29 เป็นการนำข้อมูลของหนึ่งอักขระมาแสดงบนจอภาพ โดยใช้รหัสแอสกีของอักขระมาเป็นค่าตรวจสอบตำแหน่งของข้อมูล

5. บรรทัดที่ 24-25 กำหนดค่าคงที่ของค่าประจำบิตของข้อมูลหนึ่งไบต์ เพื่อใช้ในการตรวจสอบการลงจุดให้ตรงกับข้อมูลของแต่ละอักขระ

6. บรรทัดที่ 26 ค่าคงที่เพื่อให้ข้อมูลแต่ละไบต์ในแนวตั้งของอักขระ (ข้อมูลแนวตั้งจะมี 3 ไบต์) โดยไบต์แรก (ลำดับที่ 1, 4, 7...) เริ่มที่แถวที่ 0 ไบต์ที่สอง (ลำดับที่ 2, 5, 8...) เริ่มจุดที่แถวที่ 8 และไบต์ที่สุดท้าย (ลำดับที่ 3, 6, 9...) เริ่มจุดที่แถวที่ 16

7. บรรทัดที่ 29 ทำการแสดงข้อมูลทั้งหมดของหนึ่งอักขระ โดยเริ่มที่ลำดับที่ 1 ไปสิ้นสุดที่ข้อมูลลำดับที่ 54

8. บรรทัดที่ 31-36 เป็นของการลงจุดตามข้อมูลที่ตรวจพบ โดยใช้ข้อมูลอักขระลำดับที่ตามรหัสแอสกีลบด้วย 32 คือ FontP[NoFont][ASCII-32] เมื่อ Nofont คือส่วนกำหนดฟอนต์ตัวอักขระ ดังนั้นข้อมูลที่กำหนดว่า FontP[NoFont][ASCII-32][k] คือข้อมูลหนึ่งไบต์ในลำดับที่ k ของหนึ่งอักขระนั่นเอง

9. บรรทัดที่ 33-35 เป็นการตรวจสอบว่าข้อมูลหนึ่งไบต์นั้น มีบิตใดบ้าง

รูปที่ 4 ชุดคำสั่งในการแสดงข้อความขนาดใหญ่ลงบนจอภาพ

```
40: PROCEDURE OutBig(column,row:integer; content:string);
41: var k,xpix,Ypix,c:integer;
42: begin
43:  if Content[1]<>#17 then
44:   begin
45:     column:=(column-1)*18;  row:=(row-1)*36+10;
46:   end
47:    else delete(Content,1,1);
48:  Content:=Content+' ';
49:  for k:=1 to length(content) do
50:   begin
51:   repeat
52:      if content[k] in [#28,#29] then
53:        begin
54:          NoFont:=ord(content[k])-27;
55:          inc(k);
56:        end;
57:      if content[k]=#3 then
58:        begin
59:         inc(k);  c:=ord(content[k])-48;  inc(k);
60:         if content[k]=#3 then
61:           begin
62:             inc(C,9); Inc(k);
63:           end;
64:       set_color(c);
65:      end;
66:    if content[k] in [#1..#2,#4..#27,#30,#31] then Inc(k);
67:   until content[k]>#31;
68:    xpix:=0;  Ypix:=0;
69:    if content[k]=#211 then
70:      begin
71:        ShowCharBig(column-14,row-20,211);
72:        content[k]:=#210;
73:      end;
74:    if content[k] in [#209,#212..#219,#231..#238] then xpix:=16;
75:    if (content[k] in [#232..#237]) then
76:      begin
77:        ypix:=-28;
78:        if content[k-1] in [#32..#206,#216,#217] then ypix:=-20;
79:        if content[k+1]=#211 then ypix:=-28;
80:      end;
81:    if (content[k] in [#209,#212..#215,#218,#219]) then ypix:=-20;
82:    if (content[k] in [#209,#212..#215,#218,#219]) and
83:                          (NoFont=2) then dec(xpix,2);
84:    if (content[k] in [#231..#238]) and
85:                          (NoFont=2) then dec(xpix,4);
86:    if (content[k] in [#216,#217]) then Ypix:=20;
87:     ShowCharBig(column-xpix,row+ypix,ord(content[k]));
```

รูปที่ 4 (ต่อ)

```
88:    if content[k] in [#32..#208,#210..#211,#220..#230,
89:                      #239..#255] then  inc(column,16);
90:   end;
91: end;

92: {********************* TISBIG.TPU ***************************}
93: begin
94:  ReadP24;
95: end.                                                    ■
```

ที่มีค่าไม่เป็น 0 โดยการนำไป and กับค่าคงที่ BitVal[B] มีค่าเท่ากับ BitVal [B] ก็ทำการลงจุด ณ ตำแหน่งที่คำนวณได้ โดยลงครั้งละหนึ่งจุดด้วยสีที่กำหนดไว้ก่อนหน้านี้

10. บรรทัดที่ 37 เมื่อได้ลงจุดครบ 3 ไบต์แล้ว (ค่า k mod 3=0) ก็จะเลื่อนค่า x อีกหนึ่ง ซึ่งจะมีผลข้อมูล 3 ไบต์ถัดมาแสดงต่อไปทางขวามืออีก 24 จุดในแนวตั้ง

เมื่อได้ทำงานครบถ้วนตามชุดคำสั่งนี้แล้ว ก็จะได้ภาพของอักขระปรากฏบนจอภาพได้ตามต้องการ ขั้นต่อไปจะเป็นการนำข้อความที่กำหนดให้มาแสดงผลบนจอภาพให้ถูกต้องตามอักขระวิธีในภาษาไทย คือชุดคำสั่ง OutBig ในรูปที่ 4 ซึ่งรายละเอียดมีดังนี้

1. บรรทัดที่ 43-47 เป็นการตรวจสอบว่าจะแสดงข้อความที่ตำแหน่งจากการคำนวณ หรือแบบกำหนดตำแหน่งแน่นอน คือ ถ้าอักขระตัวแรกไม่ใช่ #17(^Q) ก็จะคำนวณตำแหน่งตามคำสั่งบรรทัดที่ 45 ถ้าใช่จะใช้ค่า column, row ที่กำหนดให้มาเลย (พร้อมกับลบอักขระตัวแรกนี้ออกไปด้วย)

2. บรรทัดที่ 48 เพิ่มข้อความที่จะแสดงด้วยช่องว่างหนึ่งตัว ทั้งนี้เพราะช่วงต่อไปจะมีการตรวจรหัสแอสกีที่อยู่ถัดไปอีกหนึ่งตำแหน่งด้วย

3. บรรทัดที่ 52-56 การตรวจสอบการกำหนดฟอนต์ที่จะนำไปแสดงผลบนจอภาพ ซึ่งใช้รหัสแอสกี 28 และ 29 ควบคุม

4. บรรทัดที่ 57-65 เป็นการตรวจการแสดงสีของข้อความ โดยใช้ ^C(#3) ตามด้วยตัวเลข 0-9 ควบคุม ถ้าปิดท้ายตัวเลขด้วย ^C อีกครั้งจะได้สีที่มีค่าเพิ่มขึ้นอีก 9 เช่น #3'1'#3 คือสีหมายเลข 10

5. บรรทัดที่ 67 การทำงานในส่วนนี้จะสิ้นสุดเมื่อตรวจพบว่าอักขระมีค่ารหัสแอสกีมากกว่า 31

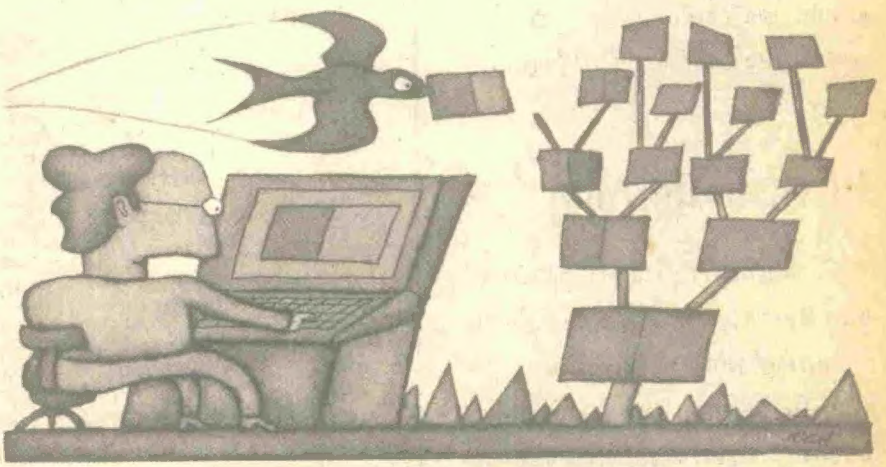
6. บรรทัดที่ 68 กำหนดตัวแปร xpix, ypix ให้มีค่าเป็น 0 โดยที่ทั้งสองค่านี้จะนำไปกำหนดเพื่อแสดงตำแหน่งของสระและวรรณยุกต์ให้ถูกต้องตามอักขระวิธีในภาษาไทย

7. บรรทัดที่ 69-73 ถ้าเป็นสระ ำ ให้แสดงอักขระนี้ที่บนอักขระที่เขียนไว้ก่อนหน้านี้ พร้อมกับกำหนดให้อักขระนี้เป็นสระ า เพื่อให้แสดงผลในคำสั่งถัดไป

8. บรรทัดที่ 74 ถ้าเป็นสระบนก็กำหนดให้ xpix+16 ซึ่งหมายถึงการแสดงอักขระนี้ที่ด้านบนของอักขระที่ได้แสดงไปก่อนหน้านี้

9. บรรทัดที่ 75-80 การคำนวณในกรณีที่พบว่าเป็นวรรณยุกต์จะกำหนดให้ค่า ypix=−28 เพื่อแสดงที่ตำแหน่งสูงจากอักขระระดับกลางไป 28 จุด แต่หากตรวจพบว่าอักขระที่อยู่ก่อนหน้านี้ไม่ใช่สระบน ก็ลดค่า ypix ให้เหลือ −20 ซึ่งจะทำให้แสดงอักขระในตำแหน่งที่ชิดกับอักขระมากขึ้น แต่ถ้าอักขระที่อยู่ถัดไปเป็นสระ ำ ก็จะกลับไปใช้ค่า ypix−28 แทน ซึ่งจะช่วยให้ข้อความที่แสดงสวยงามขึ้น ดังนั้น การป้อนข้อมูลตามแนวคิดนี้จึงต้องป้อนวรรณยุกต์ก่อนสระ ำ เสมอ

10. บรรทัดที่ 81-83 ตรวจการแสดงสระบน ั ็ ์ เพื่อให้แสดงในตำแหน่งที่ถูกต้อง ถ้าพบว่าเป็นฟอนต์ที่ 2 (ตัวเอียง) ก็จะเลื่อนการแสดงผล



ในแกน x ไปข้างหน้าอีกเล็กน้อยเพื่อให้ได้ตัวเอียงที่สวยงาม

11. บรรทัดที่ 84-85 ตรวจตำแหน่งในแกน x ของวรรณยุกต์ของฟอนต์ที่ 2

12. บรรทัดที่ 86 คำนวณตำแหน่งของสระ ุ ู ซึ่งจะแสดงด้านล่างของอักขระที่แสดงไปก่อนหน้า

13. บรรทัดที่ 87 เรียกใช้ชุดคำสั่ง ShowCharBig เพื่อแสดงอักขระที่กำหนดบนจอภาพ

14. บรรทัดที่ 88 ถ้าอักขระที่แสดงผลบนจอภาพนั้น เป็นอักขระระดับกลาง ก็จะทำการเลื่อนจุดเริ่มต้นของอักขระตัวต่อไปให้ขยับไปทางขวามือ 16 จุด (หรือ 16 แนวตั้ง)

15. บรรทัดที่ 89 จบยูนิตด้วยการอ่านข้อมูลจากไฟล์ *.p24 เข้ามาไว้ในหน่วยความจำด้วยคำสั่งตามบรรทัดที่ 94

การเก็บโปรแกรมในรูปที่ 3 และ 4 รวมกันในไฟล์ชื่อ TISBIG.PAS แล้วคอมพายล์ลงแผ่นบันทึก ก็จะได้ยูนิตไฟล์ชื่อ TISBIG.TPU ที่พร้อมจะนำไปใช้งานได้ต่อไป

หากผู้อ่านประสงค์จะใช้ยูนิตนี้เพียงยูนิตเดียว โดยไม่อาศัยยูนิต thaitis.tpu ก็ให้นำเอาชุดคำสั่งชื่อ opengraph, set_color และ readchar มารวมกับยูนิตนี้ก็จะนำไปใช้งานแบบส่วนตัวได้

โปรแกรมจ่าหน้าพัสดุ

เพื่อเป็นตัวอย่างการใช้งานของยูนิต ReadST และ TISBig จึงขอเขียนโปรแกรมตัวอย่างมาให้หนึ่งโปรแกรม โดยเป็นโปรแกรมพิมพ์จ่าหน้าพัสดุที่จัดส่งทางไปรษณีย์ เพื่อให้ได้ข้อความ

```
 1: uses crt,dos,graph,ThaiTIS,ReadST,TISBig;
 2: var Name,Add1,Add2,Add3,Add4,PostCode:string;
 3:     PrintNo:byte;

 4: PROCEDURE CopyByte(ByteSt :string);
 5: var  B:byte;  Regs:Registers;
 6: begin
 7:     for b:=1 to length(ByteSt) do
 8:      begin
 9:       Regs.AH := 0;        Regs.AL := ord(ByteSt[b]);;
10:       Regs.DX := Pred(PrintNo);  Intr($17, Regs);
11:      end;
12: end;

13: FUNCTION UP4_TOByte(B:byte):Byte;
14: const BB:array[4..7] of byte=(3,12,48,192);
15:       bck:array[4..7] of byte=(16,32,64,128);
16: var k,Bval:byte;
17: begin
18:    BVal:=0;
19:    for k:=4 to 7 do
20:      if B and Bck[k]=Bck[k] then Bval:=Bval+BB[k];
21:    UP4_TOByte:=Bval;
22: end;

23: FUNCTION DOWN4_TOByte(B:byte):Byte;
24: const BB:array[0..3] of byte=(3,12,48,192);
25:       bck:array[0..3] of byte=(1,2,4,8);
26: var k,Bval:byte;
27: begin
28:    BVal:=0;
29:    for k:=0 to 3 do
30:      if B and Bck[k]=Bck[k] then Bval:=Bval+BB[k];
31:    DOWN4_TOByte:=Bval;
32: end;
```

รูปที่ 5 ส่วนหัวของโปรแกรมพิมพ์จ่าหน้าพัสดุ และชุดคำสั่งขยายข้อมูลหนึ่งไบต์

รูปที่ 6 ชุดคำสั่งออกจอภาพด้วยเครื่องพิมพ์ 24 หัวเข็ม

```
33: PROCEDURE HardCopy24;   {ส่วนออกจอภาพสำหรับเครื่องพิมพ์ 24 หัวเข็ม }
34: var Endscan,Scanline,n1,n2:word; Mode:char; ColorOld:byte;

35:     PROCEDURE CopyDot;
36:       const
37:         Bits : array[0..7] of byte = (128,64,32,16,8,4,2,1);
38:       var
39:         Bit,PrintByte: byte; MaxX:word;
40:         XYPixel: integer;
41:     begin
42:         MaxX:=GetMaxX+1;
43:         Repeat
44:             Dec(MaxX); PrintByte:=0;
45:             for Bit := 0 to 7 do
46:                 if (GetPixel(MaxX,Scanline*8+bit) <> 0) then
47:                     PrintByte := PrintByte+ Bits[Bit];
48:         until (PrintByte<>0) or (MaxX=0);
49:       if MaxX<>0 then     begin
50:         n1 := Lo(Succ(MaxX));  n2 := Hi(Succ(MaxX));
51:         copyByte(#27'3'#16#27'*'#38+chr(n1)+chr(n2));
```

รูปที่ 6 (ต่อ)

```
52:            for XyPixel:=0 to MaxX do
53:              begin  PrintByte := 0;
54:                for Bit := 0 to 7 do
55:                  if  (GetPixel(XyPixel,Scanline*8+bit) <> 0) then
56:                        PrintByte := PrintByte+ Bits[Bit];
57:                CopyByte(chr(UP4_TOByte(PrintByte)));
58:                CopyByte(chr(DOWN4_TOByte(PrintByte))+#0);
59:              end;            end;
60:               CopyByte(#10);
61:     end;

62: BEGIN
63:    Endscan:=GetMaxY;  CopyByte(#27'1'#2);
64:    ScanLine:=0;
65:     Repeat
66:       CopyDot;  Inc(Scanline);
67:     until ScanLine>(EndScan div 8);
68:   Copybyte(#27#2#27'1'#0#27'<');
69: END;
```

```
70: {================ POSTPNT.PAS ================}
71: begin
72:  PrintNo:=1;
73:  setlinestyle(0,$FFFF,3);  set_color(13);
74:  rectangle(1,1,getmaxx-1,getmaxy-1);
75:  OutBig(160,25,#17'โปรแกรมพิมพ์จ่าหน้าพัสดุ');
76:  OutThaixy(2,5,#3'3ให้ป้อนรายละเอียดต่อไปนี้');
77:   OutThaixy(2,7,'          ชื่อ :');
78:   OutThaixy(2,9,'บ้านเลขที่ ถนน :');
79:  OutThaixy(2,11,'               :');
80:  OutThaixy(2,13,'               :');
81:  OutThaixy(2,15,'               :');
82:  OutThaixy(2,17,'  รหัสไปรษณีย์ :');
83:  Name:=ReadSTR(16,7,60,'T');
84:  Add1:=ReadSTR(16,9,60,'T');
85:  Add2:=ReadSTR(16,11,60,'T');
86:  Add3:=ReadSTR(16,13,60,'T');
87:  Add4:=ReadSTR(16,15,60,'T');
88:  Postcode:=ReadSTR(16,17,5,'E');
89:  ClsGR(0,0,getmaxx,getmaxy);
90:  setfillstyle(10,getcolor);   setlinestyle(0,$FFFF,3);
91:  bar(3,270,getmaxx,getmaxy);
92:  Rectangle(1,1,getmaxx-1,Getmaxy-1);
93:   line(0,270,getmaxx,270);
94:  OutBig(2,1,'ส่ง');
95:  OutBig(5,2,Name);
96:  OutBig(5,3,Add1);
97:  OutBig(5,4,Add2);
98:  OutBig(5,5,Add3);
99:  OutBig(5,6,Add4);
100:  OutBig(5,7,PostCode);
101:  OutBig(50,284,#17#29'จาก อาจหาญ สัตยารักษ์ ผู้ ปณ.34');
102:  OutBig(50,320,#17'     อ.เมือง ลำปาง 52000');
103:  HardCopy24;
104:  readchar;
105:  closegraph;
106: end.
```

รูปที่ 7 เป็นการป้อนข้อความเพื่อนำไปจ่าหน้าพัสดุไปรษณีย์

ที่มีขนาดโตกว่าปกติ ซึ่งใช้การป้อนข้อความที่ต้องการจ่าหน้าผ่านแป้นพิมพ์ด้วยฟังก์ชัน ReadSTR และนำข้อความที่ป้อนนี้ไปแสดงบนจอภาพด้วยคำสั่ง OutBig ของยูนิต TISBig และทำการลอกภาพหน้าจอลงกระดาษด้วยแนวการพิมพ์อีกแบบหนึ่ง จากเครื่องพิมพ์ 24 หัวเข็ม

โดยขอเริ่มที่การพิมพ์กราฟิกของเครื่องพิมพ์ 24 หัวเข็มในแนวคิดใหม่ก่อน โดยขอเขียนถึงแนวคิดในการขยายข้อมูลหนึ่งไบต์ออกเป็นข้อมูลสองไบต์ก่อน เพราะว่าตามแนวคิดนี้ผู้เขียนจะพิมพ์ผ่าน 16 หัวเข็มพิมพ์เท่านั้น (คือกำหนดให้ 8 หัวเข็มล่างไม่ทำงาน)

ในซอฟต์เมล์ไฟล์นี้จะมีชุดคำสั่ง hardcopy ซึ่งใช้ลอกจอภาพด้วยเครื่องพิมพ์ 9 หัวเข็มแถมมาให้ด้วยแล้ว จึงขอให้เรียกใช้ชุดคำสั่งให้ตรงกับเครื่องพิมพ์ที่ใช้ด้วย

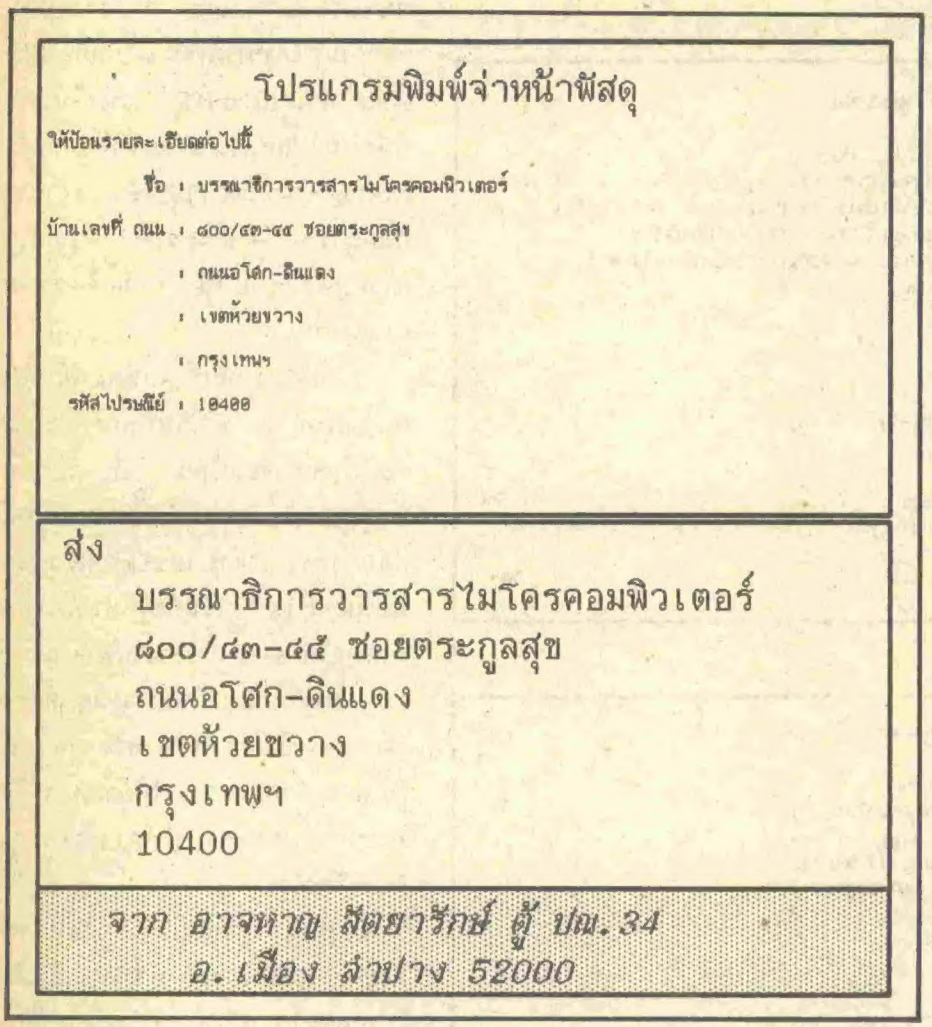
ชุดคำสั่งสำหรับการลอกจอภาพด้วยเครื่องพิมพ์ 24 หัวเข็ม เป็นดังโปรแกรมในรูปที่ 5 และรูปที่ 6 ซึ่งมีรายละเอียดดังนี้

1. บรรทัดที่ 1 กำหนดยูนิตที่ต้องนำมาใช้งานร่วมกัน

2. บรรทัดที่ 2-3 กำหนดตัวแปรเพื่อรองรับข้อมูลเกี่ยวกับชื่อ ที่อยู่ และรหัสไปรษณีย์ ส่วน PrintNo คือหมายเลขของเครื่องพิมพ์ (กำหนด 1 หรือ 2)

3. บรรทัดที่ 4-12 ชุดคำสั่งนำข้อมูลที่ได้สั่งให้เครื่องพิมพ์ทำงาน

4. บรรทัดที่ 13-22 ฟังก์ชันในการเปลี่ยนข้อมูลหนึ่งไบต์ โดยนำสี่บิตสูงมาเปลี่ยนข้อมูลเต็มหนึ่งไบต์ซึ่งใช้ค่าคงที่ BB, Bck เป็นตัวกำหนดการเปลี่ยนค่าใหม่ดังนี้ 16 เปลี่ยนเป็น 3, 32 เปลี่ยนเป็น 12, 64 เปลี่ยนเป็น 48 และ



รูปที่ 8
ภาพที่ได้จากการลอกจอภาพ (2 ครั้ง)
ด้วยเครื่องพิมพ์ 24 หัวเข็ม

128 เปลี่ยนเป็น 192 ซึ่งได้ข้อมูลเต็มหนึ่งไบต์พอดี โดยกระทำกับบิตที่ 4-7 เท่านั้น

5. บรรทัดที่ 23-32 ฟังก์ชันในการเปลี่ยนข้อมูลหนึ่งไบต์ โดยนำสี่บิตต่ำมาเปลี่ยนข้อมูลเต็มหนึ่งไบต์ ซึ่งใช้ค่าคงที่ BB, Bck เป็นตัวกำหนดการเปลี่ยนค่าใหม่ดังนี้ 1 เปลี่ยนเป็น 3, 2 เปลี่ยนเป็น 12, 4 เปลี่ยนเป็น 48 และ 8 เปลี่ยนเป็น 192 ซึ่งได้ข้อมูลเต็มหนึ่งไบต์ และเป็นการกระทำกับบิตที่ 0-3

6. บรรทัดที่ 33-69 เป็นการลอกจอภาพด้วยเครื่องพิมพ์ 24 หัวเข็ม

7. บรรทัดที่ 42 กำหนดจำนวนข้อมูลที่จะตรวจสอบการลงภาพ

8. บรรทัดที่ 43-48 เป็นการตรวจการลงจุดนับย้อนหลังจากขวาไปซ้ายจนกว่าจะพบว่ามีการลงจุดในแนวตั้ง (จำนวน 8 จุด) จึงออกมายังการทำงานอื่นต่อไป ส่วนนี้มีไว้เพื่อให้การพิมพ์มีความรวดเร็วมากยิ่งขึ้น

9. บรรทัดที่ 50-51 ส่วนการพิมพ์ข้อมูลลงกระดาษ โดยการคำนวณตัวแปรเพื่อกำหนดให้เครื่องพิมพ์ทราบจำนวนข้อมูลที่จะพิมพ์แบบกราฟิก กำหนดการเลื่อนบรรทัดเพียง 16/180 นิ้ว ซึ่งทำให้การพิมพ์ด้วยหัวเข็มเพียง 16 หัวเข็มจะเป็นการพิมพ์ต่อเนื่องกันได้พอดี

10. บรรทัดที่ 50-58 เป็นการคำนวณหาข้อมูลของภาพที่จะส่งไปพิมพ์ ซึ่งเมื่อได้ข้อมูลหนึ่งไบต์มาแล้วก็นำไปขยายเป็นข้อมูลสองไบต์ (บรรทัดที่ 57-58) เพื่อส่งไปให้เครื่องพิมพ์ 16 หัวเข็มบนทำงาน

11. บรรทัดที่ 60 เมื่อพิมพ์จบในหนึ่งบรรทัดก็จะสั่งให้เลื่อนกระดาษ

12. บรรทัดที่ 63 กำหนดจำนวนข้อมูลสูงสุดในแนวนอน (แกน x)

13. บรรทัดที่ 65-67 วนรอบพิมพ์ภาพกราฟิกที่อ่านได้จากจอภาพ

14. บรรทัดที่ 68 กำหนดให้เครื่องพิมพ์คืนสู่สภาพปกติ

ตัวอย่างง่าย ๆ ในการทดสอบฟังก์ชันในการป้อนข้อความนั้นคือ โปรแกรมจ่าหน้าพัสดุหรือจ่าหน้าซองที่มีขนาดใหญ่ ดังโปรแกรมในรูปที่ 7